LABORATORIO 7

Programas Ada con Matrices y Listas estáticas - Básico

1. Implementa la función **esta** que, dado un valor entero N y una lista estática con valores que pueden no estar ordenados, diga si el entero está o no en el vector.
2. Implementa la función **posición** que, dado un valor entero X y una lista estática cuyos valores pueden no estar ordenados, diga en qué posición se encuentra X en el vector. Si dicho entero estuviera repetido, bastará con devolver la posición de la primera de sus apariciones; y en caso de no encontrarse en el vector, se devolverá el valor Integer'Last.
3. Implementa la función **polar** que, dado un número complejo en formato trigonométrico, nos devuelve el mismo número complejo en forma polar. Para definir exactamente el ángulo, hace falta estudiar el signo de a y b y determinar el cuadrante en el que se encuentra (como si fuera una coordenada en el plano).

$$argumento = \left|\sqrt[2]{a^2 + b^2}\right| \quad angulo = \arctan\left(\frac{b}{a}\right)$$

4. Implementa la función **son_iguales** que, dadas dos listas estáticas, devuelva true si el contenido de las dos listas es el mismo (mismos elementos en el mismo orden) y false si no.
5. Crear el procedimiento **eliminar_elemento** que, dada un entero num y una lista estática de enteros L, la modifique eliminando todas las apariciones del elemento num (si existe).
6. Implementa el procedimiento **insertar_elemento_en_pos** que, dados dos enteros *num* y *pos,* y una lista estática con espacio para un entero más, inserte en la lista el número *num* en la posición *pos* dentro de la lista, desplazando los elementos que corresponda una posición a la derecha si la posición es un número entre 1 y el numero de elementos de la lista más uno. Si la posición no está en ese rango, se ignora y se deja la lista tal y como está.
7. Implementa la función **pos_maximo** que, dada una lista estática de enteros, devuelva el valor del índice en el que se encuentra el máximo de esa lista.
8. Implementa el procedimiento **ordenar_burbuja** que, dado una lista estática de enteros, ordene sus elementos de mayor a menor.
9. Implementa el procedimiento **posición** que, dada una matriz de N x M enteros (N>0 y M>0) y un valor entero Num, devuelva la posición en la que se encuentra Num (es decir, sus coordenadas Fila y Columna). Si el entero está repetido, se devolverá la posición de la fila más alta y, si aún así hay varios, entonces la de más a la izquierda. Si Num no está en la matriz, se devolverá como posición la fila 0 y la columna 0.
10. Implementa el procedimiento **maximo** que, dada una matriz de enteros de dimensiones N x M (N,M>0), calcule el valor máximo y la posición en la que se encuentra. Si el valor máximo está repetido, devolver la posición de la primera de sus apariciones en la matriz contando por filas de arriba abajo y en cada fila de izquierda a derecha (o sea, como en el ejercicio anterior).

LABORATORIO 7

Programas Ada con Matrices y Listas estáticas - Avanzado

1. Implementa la función **trigon** que, dado un número complejo en formato polar, nos devuelve el mismo número complejo en forma trigonométrica según las fórmulas que aparecen más abajo.

$$b = argumento\ sen(angulo)$$
$$a = argumento \cos(angulo)$$

2. Implementa el procedimiento **dia_siguiente** que dada una fecha, la modifique y devuelva la fecha correspondiente al día siguiente. El programa debe tener en cuenta el número de días que tiene cada mes incluyendo los años bisiestos. Un año bisiesto es aquel que es divisible entre 4, salvo que sea secular (el último de cada siglo, que termina en 00), aunque los seculares que son divisibles entre 400 sí son bisiestos.
3. Implementa el procedimiento **Maximo** que, dados los datos pluviométricos del siglo XXI, y dos fechas de este siglo, indique cuál es el mes que más ha llovido de los que se encuentran incluidos entre esas dos fechas. En los extremos solo se cuentan los días del mes que se incluyen. Por ejemplo, Entre el 10-ENE-2011 y el 15-Ene-2011 el resultado debe ser ENERO-2001, porque solo está ese mes comprendido entre esas dos fechas. Si fuera entre 10-ENE-2011 y 15-FEB-2011, entonces se contarían las lluvias entre el 10 y el 31 de enero frente a las del 1 al 15 de febrero. En la fecha de resultado, el valor del día no tiene relevancia (solo importan los otros dos).
4. Implementa el procedimiento **Simplificar** que, a partir de una lista estática de carreteras L de tipo V_Carreteras, elimine de ella las cinco primeras que sean de peaje y las devuelva en una lista nueva en el mismo orden en que se encuentran. Se considera que una carretera es de peaje cuando tiene más de 0 km de peaje. Nota: Se penalizará el uso de listas auxiliares en la resolución del problema.
5. Implementa en Ada el procedimiento **Simplificar** que, a partir de una lista de puntos no vacía L de tipo V_Puntos, la modifique para que solo contenga un cuarto de los puntos, respetando uno sí y tres no de la lista original. Los puntos que se queden en la lista deben mantener el orden de la lista original y los puntos inicial y final siempre se habrán de mantener. También se devolverá el número de puntos restantes en la lista resultado. Nota: Se penalizará el uso de listas auxiliares. Para reducir la lista se añade al final un punto de coordenadas (-1,-1), A partir de ahí, los puntos se ignoran. Por ejemplo:
   - Si L= <p1, p2> el resultado sería <p1, p2> y **2**
   - Si L= <p1, p2, p3, p4> el resultado sería <p1, p4> y **2**
   - Si L= <p1, p2, p3, p4**,** p5**,** p6**,** p7**,** p8, p9, p10**,** p11> el resultado sería <p1, p5, p9, p11> y **4**
6. Tenemos una representación de carreteras con demasiados puntos y queremos reducir el espacio de memoria necesario para representarlas aun perdiendo algo de información. Implementa en Ada el procedimiento **Simplificar** que, a partir de una lista de carreteras L de tipo V_carreteras, la modifique eliminando para cada carretera aquellos puntos cuyas diferencias en los valores X e Y con el punto anterior sean menores que 0.001 (en ambos valores). Los puntos que se queden en la lista deben mantener el orden de la lista original y los puntos inicial y final siempre se habrán de mantener. **Nota:** Se penalizará el uso de listas auxiliares en la resolución del problema.
7. Implementa la función **dni_mayor_patrimonio** que, dada una urbanización U, devuelve el DNI de la persona de la urbanización que tiene un mayor patrimonio. Se considera que el patrimonio es la suma de los valores de los pisos que posee. Para resolver este ejercicio necesitarás guardar el patrimonio de cada uno de los diferentes propietarios que te vayas encontrando e ir actualizando su patrimonio.
8. En un sudoku siempre hay 9 fichas, cada una con los 9 números (1, 2, …9). En este ejercicio se tiene una matriz NxN (9x9) para representar un sudoku a medio rellenar. En una ficha sin rellenar, en blanco, los números en blanco se representan con el valor 0. Aparte, se dispone de una ficha,

representada por una matriz cuadrada MxM (3x3), que hay que colocar en el sudoku. Por ejemplo: Se dispone del siguiente sudoku, que tiene hueco para una ficha, y una ficha a colocar:

1	6	9	2	5	8	4	3	7
5	7	8	4	9	3	6	2	1
3	4	4	7	6	1	9	5	8
9	5	2	1	8	4	7	6	3
6	1	3	5	7	9	2	8	4
4	8	7	6	3	2	5	1	9
8	3	6	9	4	5	0	0	0
7	9	1	8	2	6	0	0	0
2	4	5	3	1	7	0	0	0

Ficha

8	3	1
9	4	7
6	5	2

Una ficha no encaja en el sudoku si algún número se repite en alguna columna, fila, o en la misma ficha. Por ejemplo, tal y como está la ficha, no encaja en el hueco del sudoku, pero también se quiere examinar si puede ser colocada girada 90, 180 o 270 grados:

1	6	9	2	5	8	4	**3**	**7**
5	7	8	4	9	3	**6**	2	**1**
3	4	4	7	6	1	**9**	**5**	8
9	5	2	1	8	4	7	6	3
6	1	3	5	7	9	2	8	4
4	8	7	6	3	2	5	1	9
8	**3**	6	9	4	5	0	0	0
7	**9**	1	8	2	6	0	0	0
2	4	**5**	3	1	7	0	0	0

Ficha

8	3	1
9	4	7
6	5	2

En la fila 7, se repiten los números 8 y 3. En la fila 8 se repiten los números 7 y 9. En la fila 9, se repiten los números 2 y 5. En la columna 7, se repiten los números 6 y 9. En la columna 8 se repiten los números 3 y 5. En la columna 9, se repiten los números 1 y 7. La ficha, tal y como está, no se puede colocar en el hueco blanco del sudoku. Pero, la ficha se puede rotar hacia la derecha para poder encontrar su colocación. Por ejemplo, si se rota la ficha hacia la derecha, se obtienen las siguientes fichas:

8	3	1
9	4	7
6	5	2

↷

6	9	8
5	4	3
2	7	1

↷

2	5	6
7	4	9
1	3	8

↷

1	7	2
3	4	5
8	9	6

Si se rota la ficha 270º, entonces sí que encaja en el sudoku, ya que los números no se repiten en ninguna fila ni columna:

1	6	9	2	5	8	4	3	7
5	7	8	4	9	3	6	2	1
3	4	4	7	6	1	9	5	8
9	5	2	1	8	4	7	6	3
6	1	3	5	7	9	2	8	4
4	8	7	6	3	2	5	1	9
8	3	6	9	4	5	0	0	0
7	9	1	8	2	6	0	0	0
2	4	5	3	1	7	0	0	0

Ficha

1	7	2
3	4	5
8	9	6

9. Implementa el procedimiento **encontrar_espacio_blanco**. Un espacio en blanco es una submatriz MxM con todas las casillas con el valor 0. Se supone que si un número tiene el valor 0, el resto de los números de esa ficha también lo son.
10. Implementa el procedimiento **rotar_matriz_derecha_90**, que, dada una ficha, la rote 90º a la derecha. Nota: Se busca una solución transportable a fichas de cualquier tamaño, por lo que se penalizarán soluciones que no incluyan un bucle.
11. Implementa la función **filas_correctas**, que, dados un sudoku sin terminar (con un espacio para una ficha), una ficha, y la especificación de la casilla superior izquierda de un hueco del sudoku, compruebe que si el resultado de añadir la ficha a el sudoku con un espacio en blanco, todas las filas y las columnas cumplen los requisitos